Some modelling aspects for the Matlab implementation of MMA

Krister Svanberg
krille@math.kth.se

Optimization and Systems Theory
Department of Mathematics
KTH, SE–10044 Stockholm
September 2004

1. Considered optimization problem.

The Matlab version of the author's MMA code is based on the assumption that the users optimization problem is written on the following form, where the optimization variables are $\mathbf{x} = (x_1, \ldots, x_n)^T$, $\mathbf{y} = (y_1, \ldots, y_m)^T$ and z.

$$\begin{array}{rll} \text{minimize} & f_0(\mathbf{x}) + a_0 z + \sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) & \\ \text{subject to} & f_i(\mathbf{x}) - a_i z - y_i \le 0, & i = 1, \ldots, m \\ & x_j^{\min} \le x_j \le x_j^{\max}, & j = 1, \ldots, n \\ & y_i \ge 0, & i = 1, \ldots, m \\ & z \ge 0. & \end{array} \tag{1.1}$$

Here, $x_1, \ldots, x_n$ are the "true" optimization variables, while $y_1, \ldots, y_m$ and z are "artificial" optimization variables which will be motivated below.
$f_0, f_1, \ldots, f_m$ are given, continuously differentiable, real-valued functions.
$x_j^{\min}$ and $x_j^{\max}$ are given real numbers which satisfy $x_j^{\min} < x_j^{\max}$.
a_0 and a_i are given real numbers which satisfy $a_0 > 0$ and $a_i \ge 0$.
c_i and d_i are given real numbers which satisfy $c_i \ge 0$, $d_i \ge 0$ and $c_i + d_i > 0$.

2. Ordinary NLP problems.

Assume that the user wants to solve a problem on the following "standard" form for nonlinear programming.

$$\begin{array}{rll} \text{minimize} & f_0(\mathbf{x}) & \\ \text{subject to} & f_i(\mathbf{x}) \le 0, & i = 1, \ldots, m \\ & x_j^{\min} \le x_j \le x_j^{\max}, & j = 1, \ldots, n \end{array} \tag{2.1}$$

To put (1.1) into this form (2.1), first let $a_0 = 1$ and $a_i = 0$ for all i. Then $z = 0$ in any optimal solution of (1.1). Further, for each i, let $d_i = 0$ and $c_i =$ "a large number", so that the variables y_i become "expensive". Then typically $\mathbf{y} = \mathbf{0}$ in any optimal solution of (1.1), and the corresponding $\mathbf{x}$ is an optimal solution of (2.1).

It should be noted that the problem (1.1) always has feasible solutions, and in fact also at least one optimal solution. This holds even if the user's problem (2.1) does not have any feasible solutions, in which case some $y_i > 0$ in the optimal solution of (1.1). This is just one advantage of the formulation (1.1) compared to the formulation (2.1).

Now some practical considerations and recommendations.

In many applications, the constraints are on the form $\sigma_i(\mathbf{x}) \leq \sigma_i^{\max}$, where $\sigma_i(\mathbf{x})$ stands for e.g. a certain stress, while $\sigma_i^{\max}$ is the largest permitted value on this stress. This means that $f_i(\mathbf{x}) = \sigma_i(\mathbf{x}) - \sigma_i^{\max}$ (in (1.1) as well as in (2.1)). The user should then preferably scale the constraints in such a way that $1 \leq \sigma_i^{\max} \leq 100$ for each i (and not $\sigma_i^{\max} = 10^{10}$). The objective function $f_0(\mathbf{x})$ should preferably be scaled such that $1 \leq f_0(\mathbf{x}) \leq 100$ for reasonable values on the variables. The variables x_j should preferably be scaled such that $0.1 \leq x_j^{\max} - x_j^{\min} \leq 100$, for all j.

Concerning the "large numbers" on the coefficients c_i (mentioned above), the user should for numerical reasons try to avoid "extremely large" values on these coefficients (like 10^{10}). It is better to start with "reasonably large" values and then, if it turns out that not all $y_i = 0$ in the optimal solution of (1.1), increase the corresponding values of c_i by e.g. a factor 100 and solve the problem again, etc. If the functions and the variables have been scaled according to above, then "resonably large" values on the parameters c_i could be, say, $c_i = 1000$ or 10000.

Finally, concerning the simple bound constraints $x_j^{\min} \leq x_j \leq x_j^{\max}$, it may sometimes be the case that some variables x_j do not have any prescribed upper and/or lower bounds. In that case, it is in practice always possible to choose "artificial" bounds $x_j^{\min}$ and $x_j^{\max}$ such that every realistic solution $\mathbf{x}$ satisfies the corresponding bound constraints. The user should then preferably avoid choosing $x_j^{\max} - x_j^{\min}$ unnecessarily large. It is better to try some reasonable bounds and then, if it turns out that some variable x_j becomes equal to such an "artificial" bound in the optimal solution of (1.1), change this bound and solve the problem again (starting from the recently obtained solution), etc.

3. Least squares problems. (Minimum 2–norm problems.)

Assume that the user wants to solve a constrained least squares problem on the form

$$\begin{array}{rll} \text{minimize} & \displaystyle\sum_{i=1}^{p} (h_i(\mathbf{x}))^2 & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, & i = 1, \ldots, q \\ & x_j^{min} \leq x_j \leq x_j^{max}, & j = 1, \ldots, n \end{array} \tag{3.1}$$

where h_i and g_i are given differentiable functions.

The functions f_i and the parameters a_i, c_i and d_i should then be chosen as follows in problem (1.1).

$$\begin{array}{lcll} m & = & 2p+q, & \\ f_0(\mathbf{x}) & = & 0, & \\ f_i(\mathbf{x}) & = & h_i(\mathbf{x}), & i=1,\ldots,p \\ f_{p+i}(\mathbf{x}) & = & -h_i(\mathbf{x}), & i=1,\ldots,p \\ f_{2p+i}(\mathbf{x}) & = & g_i(\mathbf{x}), & i=1,\ldots,q \\ a_0 & = & 1, & \\ a_i & = & 0, & i=1,\ldots,m \\ d_i & = & 2, & i=1,\ldots,2p \\ d_{2p+i} & = & 0, & i=1,\ldots,q \\ c_i & = & 0, & i=1,\ldots,2p \\ c_{2p+i} & = & \text{large number}, & i=1,\ldots,q \end{array}$$

4. Minimum 1–norm problems.

Assume that the user wants to solve a minimum 1–norm problem on the form

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{p} |h_i(\mathbf{x})| & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, & i=1,\ldots,q \\ & x_j^{min} \leq x_j \leq x_j^{max}, & j=1,\ldots,n \end{array} \tag{4.1}$$

where h_i and g_i are given differentiable functions. The functions f_i and the parameters a_i, c_i and d_i should then be chosen as follows in problem (1.1).

$$\begin{array}{lcll} m & = & 2p+q, & \\ f_0(\mathbf{x}) & = & 0, & \\ f_i(\mathbf{x}) & = & h_i(\mathbf{x}), & i=1,\ldots,p \\ f_{p+i}(\mathbf{x}) & = & -h_i(\mathbf{x}), & i=1,\ldots,p \\ f_{2p+i}(\mathbf{x}) & = & g_i(\mathbf{x}), & i=1,\ldots,q \\ a_0 & = & 1, & \\ a_i & = & 0, & i=1,\ldots,m \\ d_i & = & 0, & i=1,\ldots,m \\ c_i & = & 1, & i=1,\ldots,2p \\ c_{2p+i} & = & \text{large number}, & i=1,\ldots,q \end{array}$$

5. Minimax problem. (Minimum ∞–norm problems.)

Assume that the user wants to solve a minimax problem on the form

$$\begin{array}{lll} \text{minimize} & \max\limits_{i=1,..,p} \{|h_i(\mathbf{x})|\} & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, & i=1,\ldots,q \\ & x_j^{min} \leq x_j \leq x_j^{max}, & j=1,\ldots,n \end{array} \tag{5.1}$$

where h_i and g_i are given differentiable functions.

The functions f_i and the parameters a_i, c_i and d_i should then be chosen as follows in problem (1.1).

$$\begin{array}{lcll} m & = & 2p+q, & \\ f_0(\mathbf{x}) & = & 0, & \\ f_i(\mathbf{x}) & = & h_i(\mathbf{x}), & i=1,\ldots,p \\ f_{p+i}(\mathbf{x}) & = & -h_i(\mathbf{x}), & i=1,\ldots,p \\ f_{2p+i}(\mathbf{x}) & = & g_i(\mathbf{x}), & i=1,\ldots,q \\ a_0 & = & 1, & \\ a_i & = & 1, & i=1,\ldots,2p \\ a_{2p+i} & = & 0, & i=1,\ldots,q \\ d_i & = & 0, & i=1,\ldots,m \\ c_i & = & \text{large number}, & i=1,\ldots,m \end{array}$$

6. The MMA subproblem

MMA is a method for solving problems on the form (1.1), using the following approach: In each iteration, the current iteration point $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}, z^{(k)})$ is given. Then an approximating explicit subproblem is generated. In this subproblem, the functions $f_i(\mathbf{x})$ are replaced by approximating convex functions $\tilde{f}_i^{(k)}(\mathbf{x})$. These approximations are based mainly on gradient information at the current iteration point, but also (implicitly) on information from previous iteration points. The subproblem is solved, and the unique optimal solution becomes the next iteration point $(\mathbf{x}^{(k+1)}, \mathbf{y}^{(k+1)}, z^{(k+1)})$. Then a new subproblem is generated, etc. The subproblem mentioned above looks as follows.

$$\begin{array}{rll} \text{minimize} & \tilde{f}_0^{(k)}(\mathbf{x}) + a_0 z + \displaystyle\sum_{i=1}^{m}(c_i y_i + \tfrac{1}{2} d_i y_i^2) & \\ \text{subject to} & \tilde{f}_i^{(k)}(\mathbf{x}) - a_i z - y_i \le 0, & i=1,\ldots,m \\ & \alpha_j^{(k)} \le x_j \le \beta_j^{(k)}, & j=1,\ldots,n \\ & y_i \ge 0, & i=1,\ldots,m \\ & z \ge 0. & \end{array} \tag{6.1}$$

The approximating functions $\tilde{f}_i^{(k)}(\mathbf{x})$ are chosen as

$$\tilde{f}_i^{(k)}(\mathbf{x}) = \sum_{j=1}^{n}\left(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k)}}{x_j - l_j^{(k)}}\right) + r_i^{(k)}, \quad i = 0, 1, \ldots, m\,,$$

where

$$p_{ij}^{(k)} = (u_j^{(k)} - x_j^{(k)})^2 \left(\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})\right)^+ + \kappa_{ij}^{(k)}\right),$$

$$q_{ij}^{(k)} = (x_j^{(k)} - l_j^{(k)})^2 \left(\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})\right)^- + \kappa_{ij}^{(k)}\right),$$

$$r_i^{(k)} = f_i(\mathbf{x}^{(k)}) - \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j^{(k)}} + \frac{q_{ij}^{(k)}}{x_j^{(k)} - l_j^{(k)}} \right),$$

$$\alpha_j^{(k)} = \max\{x_j^{\min}, 0.9l_j^{(k)} + 0.1x_j^{(k)}\},$$

$$\beta_j^{(k)} = \min\{x_j^{\max}, 0.9u_j^{(k)} + 0.1x_j^{(k)}\}.$$

Here,

$$\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})\right)^+ = \max\{0\,,\, \frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})\} \;\text{ and }\; \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})\right)^- = \max\{0\,,\, -\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})\}.$$

The default rules for updating the lower asymptotes $l_j^{(k)}$ and the upper asymptotes $u_j^{(k)}$ are as follows. The first two iterations, when $k = 1$ and $k = 2$,

$$l_j^{(k)} = x_j^{(k)} - 0.5(x_j^{\max} - x_j^{\min}),$$

$$u_j^{(k)} = x_j^{(k)} + 0.5(x_j^{\max} - x_j^{\min}).$$

In later iterations, when $k \geq 3$,

$$l_j^{(k)} = x_j^{(k)} - \gamma_j^{(k)}(x_j^{(k-1)} - l_j^{(k-1)}),$$

$$u_j^{(k)} = x_j^{(k)} + \gamma_j^{(k)}(u_j^{(k-1)} - x_j^{(k-1)}),$$

where

$$\gamma_j^{(k)} = \begin{cases} 0.7 & \text{if} \quad (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) < 0, \\ 1.2 & \text{if} \quad (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) > 0, \\ 1 & \text{if} \quad (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) = 0. \end{cases}$$

The default values of the parameters $\kappa_{ij}^{(k)}$ are

$$\kappa_{ij}^{(k)} = 10^{-3} \cdot \left| \frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \right| \; + \; \frac{10^{-6}}{u_j^{(k)} - l_j^{(k)}}\,, \quad \text{for } \; i = 0, 1, .., m \;\text{ and }\; j = 1, .., n. \tag{6.2}$$

This implies that all the approximating functions $\tilde{f}_i^{(k)}$ are strictly convex, which in turn implies that there is always a unique optimal solution of the MMA subproblem.

Regardless of the values of the parameters $\kappa_{ij}^{(k)}$, the functions $\tilde{f}_i^{(k)}$ are always first order approximations of the original functions f_i at the current iteration point, i.e.

$$\tilde{f}_i^{(k)}(\mathbf{x}^{(k)}) = f_i(\mathbf{x}^{(k)}) \;\text{ and }\; \frac{\partial \tilde{f}_i^{(k)}}{\partial x_j}(\mathbf{x}^{(k)}) = \frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}). \tag{6.3}$$